UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Premrl

Razvoj spletne aplikacije za prikaz porabe energentov pri proizvodnji asfaltnih zmesi

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2017

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Premrl

Razvoj spletne aplikacije za prikaz porabe energentov pri proizvodnji asfaltnih zmesi

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Igor Rožanc

Ljubljana, 2017

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi predstavite razvoj spletne aplikacije za prikaz porabe energentov pri proizvodnji cementnih in asfaltnih zmesi. V ta namen podrobno preučite zahteve, izberite primerne tehnologije ter sistematično predstavite razvoj rešitve. Poudarek naj bo na uporabnosti rešitve v konkretnem okolju ter na uporabi ustreznih pristopov za nazoren grafični prikaz porabe energentov. V zadnjem delu naloge analizirajte opravljeno delo ter predlagajte izboljšave.

Zahvaljujem se mentorju viš. pred. dr. Igorju Rožancu za vse napotke pri izdelavi diplomskega dela. Prav tako se zahvaljujem gospodu Andreju Lavrenčiču, mag. gosp. inž., dipl. inž. str., in Tomažu Mlakarju, da sta mi omogočila soočenje s težavo, ki izvira iz stvarnosti. Še posebej se zahvaljujem družini, prijateljem in sošolcem za vso podporo tekom študija.

Svojemu bratu Tinetu.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CSS	Cascading Style Sheets	kaskadne slogovne podloge
HTTP	Hyper Text Transfer Protocol	protokol za prenos nadbesedil
IDE	Integrated Development Environment	integrirano razvojno okolje
API	Application Program Interface	programski vmesnik
URL	Uniform Resource Locator	enolični krajevnik vira
ORM	Object-Relational Mapping	objektno relacijsko preslikovanje
npm	Node.js package manager	upravljalnik s paketi Node.js
SVG	Scalable Vector Graphics	umerljiva vektorska grafika
VLM	Vector Markup Language	opisni jezik za prikaz vektorske grafike
AJAX	Asynchronous JavaScript and XML	asinhroni JavaScript in XML
MVC	Model View Controller	model, pogled, krmilnik
HTML	Hyper Text Markup Language	označevalni jezik za nadbesedila
JSON	JavaScript Object Notation	Objekt JavaScript za prenašanje podatkov
CRUD	Create, Read, Update, Delete	ustvari, beri, posodobi, zbriši

Povzetek

Naslov: Razvoj spletne aplikacije za prikaz porabe energentov pri proizvodnji asfaltnih zmesi

V diplomskem delu opisujemo razvoj spletne aplikacije za prikaz porabe energentov v podjetju, ki se ukvarja s proizvodnjo kamnitih agregatov, betonov in asfaltnih zmesi. Izdelana spletna aplikacija rešuje problem preglednega prikaza zbranih podatkov o porabi energentov ter olajša izračunavanje stroškov povezanih s porabo energentov. V uvodu je predstavljena motivacija ter zahteve spletne aplikacije. V glavnem delu diplomskega dela je najprej opisano novo odprtokodno programsko ogrodje ASP.NET Core, v katerem je bila aplikacija razvita. To omogoča razvoj in poganjanje spletnih aplikacij na različnih operacijskih sistemih. Za obličje smo uporabili več preizkušenih knjižnic v jeziku JavaScript. Sledi opis postopka razvoja spletne aplikacije, primeri uporabe in ovrednotenje opravljenega dela. Z uporabo sistematičnega razvoja v opisanih tehnologijah smo dobili aplikacijo, ki je odzivna, prilagodljiva, razširljiva in ima hkrati zagotovljeno dolgoročno podporo. Na koncu smo še analizirali opravljeno delo. Aplikacija je trenutno še v fazi preizkušanja, v prihodnje pa bo nameščena tudi v delovno okolje. To je prepuščeno nadaljnjemu sodelovanju med razvijalcem in uporabnikom in presega okvir diplomskega dela.

Ključne besede: spletna aplikacija, ASP.NET Core, MVC, JavaScript, prikaz porabe energentov, proizvodnja asfaltnih zmesi

Abstract

Title: Development of a web application for display of energy consumption in asphalt mixtures production

This thesis describes the development of a web application that displays energy consumption in a company that produces stone aggregates, concrete and asphalt mixtures. The developed web application offers a solution to cluttered data collection of energy consumption and alleviates calculating the costs linked to energy consumption. In the introduction we present the background of the problem, and the given requirements. In the main part of the thesis we describe the new open-source framework ASP.NET Core in which the application was developed. This framework enables the developers to build and run web applications in different operating systems. Several tested JavaScript libraries were used for the frontend. Technologies used in the development of the web application are then followed by the description of the development process, use cases and evaluation of the completed work. By using a systematic development in the described technologies we built a responsive, adaptable and extendable application with a long-term support. We concluded with an analysis of the presented work. The application is currently still in testing and it will be installed in working environment in the near future. That part is a matter of a further refinements between developer and user.

Keywords: web application, ASP.NET Core, MVC, JavaScript, display of energy consumption, asphalt mixtures production

Poglavje 1 Uvod

V okviru diplomskega dela smo dobili priložnost izdelati spletno aplikacijo, ki rešuje praktični problem resničnega podjetja. V podjetju ciljnega uporabnika aplikacije CPG, d. d., s polnim imenom Cestno podjetje Nova Gorica, družba za vzdrževanje in gradnjo cest, d. d., se ukvarjajo z vzdrževanjem in gradnjo cest. V praktičnem delu naloge smo razvijali spletno aplikacijo, ki bi nudila pomoč pri načrtovanju investicij glede na višino preteklih stroškov, vezanih na porabo energentov v asfaltni bazi Laže. V uvodnih poglavjih bomo podrobneje predstavili ozadje, motivacijo in povzetek funkcionalnosti spletne aplikacije.

1.1 Ozadje

V asfaltni bazi Laže se ukvarjajo s proizvodnjo betonov in asfaltnih zmesi. Eden izmed korakov pri pridobivanju asfaltne zmesi je gretje kamnitih zrn v sušilnem bobnu, preden se le-ta v nadaljnjih korakih zmešajo s segretim bitumnom in tako skupaj tvorijo asfaltno zmes. Trenutna konfiguracija sistema omogoča spremljanje porabe energenta v sušilnem bobnu z integriranim gorilnikom. Energent se v intervalih dovaja na gorilec. Interval je sestavljen iz treh korakov: vklopa, delovanja ter izklopa. Programabilni krmilnik beleži intervale in začetne ter končne vrednosti, ki se enkrat dnevno (po zadnjem intervalu) zapišejo v bazo. Podatki se prenašajo po lokalnem omrežju z uporabo komunikacijskega protokola Modbus [1]. Za opazovanje trenutnega stanja v gorilcu imajo namensko programsko opremo, za pregled podatkov v podatkovni bazi pa je nimajo, zato so podatki dostopni le njihovemu informatiku.

1.2 Motivacija

Velik delež stroškov obrata predstavljajo energenti. Poraba in cena le-teh zelo niha, zato želijo vodstveni kadri imeti možnost pregleda nad porabo in gibanjem cen preko enostavnega uporabniškega vmesnika. Izračuni namreč trenutno potekajo ročno, kar je zamudno in nepraktično. Kot smo pojasnili v prejšnjem podpoglavju, imamo trenutno na voljo zbrane podatke za porabo energenta v gorilniku. Te podatke in podatke o cenah bi radi združili na pregleden način. Na ta način bi lažje predvideli tudi prihodnjo porabo in tako načrtovali morebitne investicije na podlagi gibanja stroškov. Ker želimo dostop do teh podatkov

kadarkoli in na kakršnikoli napravi, je smiselno, da to rešimo s spletno aplikacijo, ki se prilagaja glede na velikost zaslona naprave.

1.3 Rešitev

Na trgu obstaja več rešitev, ki omogočajo spremljanje porabe energentov. Rešitve ponujajo podjetja kot so Top Teh d. o. o. [2], Elektro Maribor d. d. preko spletno aplikacije eStoritev [3], Efergy Technologies Ltd. s produktom Engage [4] ter podjetje Current Cost Ltd. [5] z raznimi produkti. Nekatere izmed rešitev podpirajo tudi upravljanje z energetskim proračunom. Pri tem smo opazili več pomanjkljivosti, kot so na primer previsoka cena investicije, posegi v naprave, ki niso v neposredni lasti podjetja (primer električnih omaric, ki so v lasti distributerja električne energije), omejenost na določen energent, vezanost na določenega ponudnika energentov, omejen nabor platform, ki podpirajo aplikacijo, nepopolnost izvedbe in shranjevanje potencialno občutljivih podatkov podjetja v oblak. Zaradi naštetih vzrokov smo se odločili za izdelavo lastne rešitve.

V okviru diplomskega dela bomo razvili spletno aplikacijo, ki vsebuje naslednje funkcionalnosti:

- pregledovanje porabe energentov za izbrano časovno obdobje,
- izračun in prikaz stroškov za izbrano časovno obdobje,
- pregledovanje časovnega deleža uporabe gorilnika za izbrano časovno obdobje,
- pregledovanje gibanja cen za izbrano časovno obdobje,
- vnos in urejanje cene energenta,
- pregledovanje dostopov v sistem za izbrano časovno obdobje,
- registracija in urejanje uporabnikov,
- urejanje uporabniškega računa.

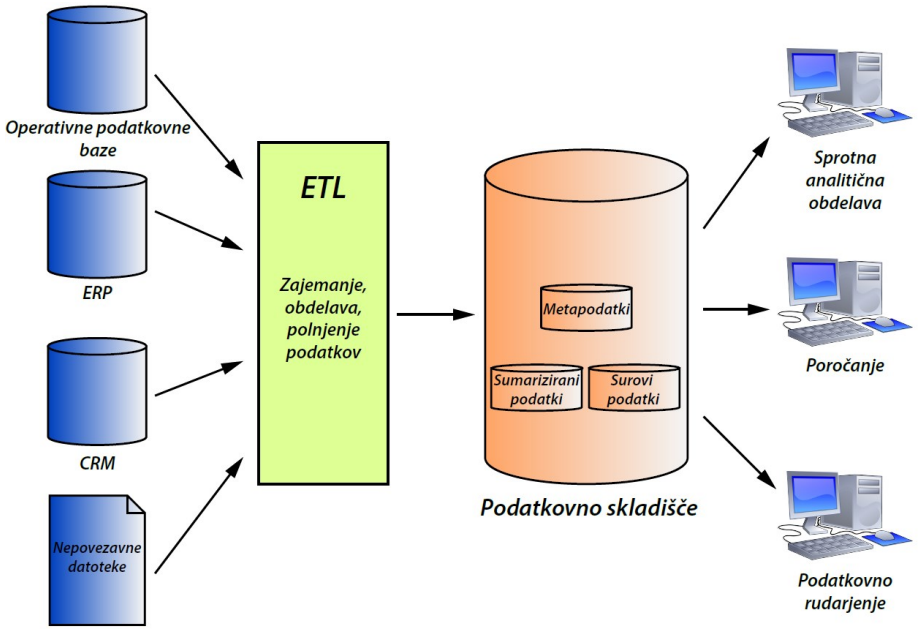
Spletno aplikacijo lahko uporabljajo le registrirani uporabniki. Spletna aplikacija je zasnovana tako, da so določene funkcionalnosti omogočene glede na vlogo uporabnika.

Podatke v primerni obliki zagotavljajo na strani uporabnika, aplikacija pa bo nameščena na strežnik, ki bo imel lokalni dostop do njihove podatkovne baze. Zaradi splošnosti izvedbe bodo lahko aplikacijo uporabljala tudi druga podjetja in organizacije, ki lahko zagotovijo

primeren dostop in obliko vhodnih podatkov. Poleg zgoraj naštetih funkcionalnosti ni bilo podanih nobenih drugih nefunkcionalnih zahtev.

1.4 Primerjava z drugimi rešitvami

Običajno za podporo poslovnemu odločanju uporabljamo podatkovna skladišča [6]. Podatkovno skladiščenje s svojimi strukturami, v podporo poslovnemu procesu prirejenimi postopki, in v procesu migracije podatkov prečiščenimi ter integriranimi podatki omogoča celovit pogled na delovanje posamezne organizacije. Slika 1.1 zajema vse lastnosti dobro zasnovanega in izvedenega podatkovnega skladišča.



Slika 1.1: Shema podatkovnega skladišča

Podatkovno skladišče naj bi ustrezalo naslednjim kriterijem:

- vsebinska organizacija po entitetah podjetja,
- integriteta podatkov iz več aplikativnih sistemov v okviru IS podjetja in omogočanje konsistentnega pregleda nad podatki,
- podatki hranijo časovno komponento, kar nam omogoča opazovanje trendov,
- podatki niso podvrženi spremembam v realnem času s strani aplikacij.

Uporaba podatkovnih skladišč zahteva uporabnike s primernimi računalniškimi znanji, kar je lahko v praksi težava. Tudi zato si v okviru diplomskega dela nismo zadali za cilj tako visoke

stopnje funkcionalnosti in smo se odločili za izvedbo le preprostejših pregledov podatkov in enostavnih izračunov za preteklo obdobje brez na primer izdelovanja poročil za prihodnost in podobnega.

V naslednjih poglavjih sledi razlaga in utemeljitev izbire tehnologij za izvedbo rešitve, opis razvoja rešitve in prikaz uporabniškega vmesnika, analiza naše rešitve ter končni sklep.

Poglavje 2 Uporabljena orodja in tehnologije

V tem poglavju bomo predstavili orodja in tehnologije, ki smo jih uporabili pri razvoju spletne aplikacije. Orodja in tehnologije smo izbrali na podlagi uveljavljenosti, aktualnosti, priporočil izkušenejših razvijalcev in predvsem brezplačnosti. Z nekaterimi izmed njih smo že imeli izkušnje, kar je povečalo učinkovitost izdelave rešitve.

2.1 Orodja

2.1.1 Visual Studio Community 2015

Visual Studio Community 2015 [7], [8] je integrirano razvojno okolje podjetja Microsoft. Samostojni razvijalci lahko IDE uporabljajo za razvoj tako brezplačnih kot tudi plačljivih aplikacij. IDE lahko brezplačno uporablja neomejeno število uporabnikov znotraj organizacije v učnem okolju, za raziskovalne namene akademske narave ali za razvoj prosto dostopnih projektov. V nepridobitnih organizacijah se lahko uporablja IDE tudi za druge namene, če število uporabnikov ne presega števila pet. Za organizacije pridobitne narave, če število uporabnikov presega 250 ali pa njihov letni promet presega en milijon ameriških dolarjev, je uporaba plačljiva.

Urejevalnik kode podpira komponento IntelliSense, ki pomaga pri pisanju programov z mehanizmi za samodejno dopolnjevanje in preurejanje kode. V okolje je integriran tudi razhroščevalnik, ki nam pomaga pri iskanju napak in spremljanju vrednosti spremenljivk na nivoju izvorne ter na nivoju strojne kode. Na voljo imamo tudi sistem za nadzor različic, ki je kompatibilen s sistemom za nadzor različic Git [9].

Okolje podpira razvoj v različnih programskih jezikih za različne platforme, od platform za mobilne naprave preko spletnih aplikacij do aplikacij, ki delujejo na različnih operacijskih sistemih.

2.1.2 Chrome Developer Tools

Chrome Developer Tools [10] je skupek orodij za spletni razvoj. Orodja ponujajo razvijalcem vpogled v delovanje brskalnika in delovanje spletnih aplikacij. S pomočjo teh orodij lahko učinkovito poiščemo napake pri postavitvi strani, odkrivamo napake v programski kodi JavaScript in optimiziramo programsko kodo. Orodja so razvrščena v osem osnovnih skupin kot so na primer elementi spletne strani, viri, omrežje, konzola, časovnica in tako dalje.

2.1.3 Postman

Postman [11] je razvojno orodje, ki nam preko grafičnega vmesnika pomaga pri grajenju, testiranju in dokumentiranju spletnih programskih vmesnikov. Obnaša še kot spletni odjemalec in nam omogoča vnašanje in spremljanje zahtevkov HTTP. Prikaže nam telo odgovora na določeno zahtevo, piškotke in metapodatke. Spremlja tudi zgodovino zahtevkov in v navezi s spletnim brskalnikom Google Chrome omogoča zajemanje piškotkov. Tako lahko recimo simulira stanje uspešne prijave v določeno spletno aplikacijo.

2.1.4 Microsoft Access

Microsoft Access [12] je sistem za upravljanje s podatkovnimi bazami podjetja Microsoft. Združuje strežnik Microsoft Jet Database Engine [13] z grafičnim uporabniškim vmesnikom in pripomočki za razvoj računalniških programov. Vključen je v programski paket Microsoft Office.

Microsoft Access shranjuje podatke v lastnem formatu Access Jet Database Engine, omogoča pa tudi uvoz ali povezovanje s podatki iz drugih podatkovnih baz ali aplikacij.

Razvoj temelji na uporabi objektnega programskega jezika Visual Basic for Applications.

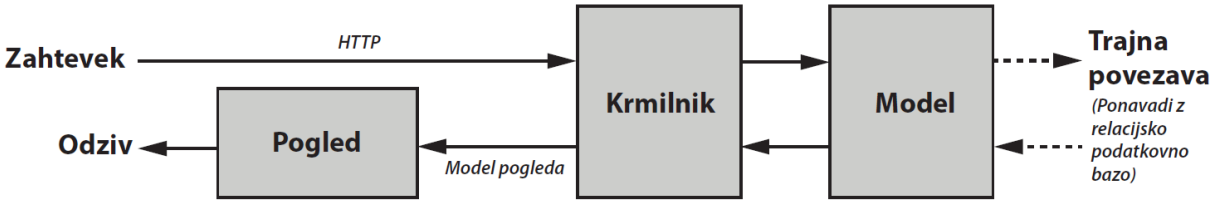
2.2 Tehnologije

2.2.1 Ogrodje ASP.NET Core

ASP.NET Core [14] je Microsoftovo ogrodje za razvoj programske opreme. Omogoča razvoj aplikacij in servisov na različnih napravah v različnih operacijskih sistemih. Združuje zmožnosti načela MVC [15] in vmesnika Web API [16] v eno samo spletno programsko ogrodje. Skoraj vse zmožnosti sistema so implementirane po načelu modularnosti, ki jih dodamo v rešitev v obliki paketov NuGet [17]. Posledica tega so manjši, bolj varni in obvladljivi programi, ki so v koraku z modernimi smernicami pri razvoju spletnih aplikacij.

2.2.1.1 ASP.NET Core MVC

MVC [15] je arhitekturno načelo, ki loči aplikacijo na tri glavne komponente: model, pogled ter krmilnik. S pomočjo tega načela dosežemo delitev odgovornosti. Uporabnikovo rokovanje z aplikacijo, ki sledi načelu MVC, sledi naravnemu vzorcu. Glede na uporabnikovo akcijo aplikacija reagira tako, da spremeni podatke v model, nato pa uporabniku vrne posodobljen pogled. Spletne aplikacije v ta namen uporabljajo kombinacijo različnih elementov (kot so podatkovne baze, HTML, izvršljiva koda itd.), ki so po navadi razvrščene v množico plasti in te se naravno preslikajo po načelu MVC. Slika 2.1 prikazuje shemo načela MVC v ogrodju ASP.NET Core.



Slika 2.1: Shema MVC v ogrodju ASP.NET Core.

2.2.1.2 Usmerjanje

Komponenta za usmerjanje v ASP.NET Core MVC nam omogoča izdelavo aplikacij, ki imajo uporabniku bolj razumljive naslove URL. Ti hkrati tudi sledijo vzorcu, ki pripomore k boljšemu pozicioniranju na iskalnih straneh.

Obstajata dva načina usmerjanja. Prvi se imenuje *usmerjanje po konvenciji*, s katerim lahko definiramo format naslovov URL na globalni ravni. Po načinu *usmerjanja z uporabo atributov* pa pot definiramo v vrednosti atributa nad določenim krmilnikom ali akcijo.

2.2.1.3 Vezave na modele

Vezava na modele nam omogoča pretvarjanje podatkov iz zahtevka HTTP v objekte, s katerimi lahko krmilnik operira. Krmilnik tako ne potrebuje dodatnih preverjanj, da bi ugotovil, kateri podatki se preslikajo v ustrezne parametre metode, ki predstavlja akcijo.

2.2.1.4 Preverjanje veljavnosti modela

Preverjanje veljavnosti modela poteka z definiranimi pravili, ki se uporabljajo tako, da jih dodamo lastnostim v obliki atributov. Preverjanje se zgodi pred pošiljanjem sprememb na strežnik, dodatno pa tudi na strežniku pred klicem akcije.

2.2.1.5 Vrinjena odvisnost

Krmilniki lahko prosijo za potrebne storitve preko konstruktorja in na ta način sledijo načelu, da objekt jasno izrazi svoje potrebe do drugih objektov. Tako so mu te storitve na voljo v trenutku, ko jih želi uporabiti.

2.2.1.6 Filtri

Filtri se uporabljajo zato, da olajšajo razvijalcu programske opreme avtorizacijo in upravljanje z izjemami. Taka opravila so odvisna ali vplivajo na več delov sistema in tako ne sodijo v koncept objektno usmerjenega ali postopkovnega programiranja.

2.2.1.7 Območja

Po konvenciji nam območja omogočijo grupiranje v manjše, smiselno povezane skupine. Tako imamo logične komponente, kot so model, krmilnik in pogled, razvrščene vsako v svoj imenik.

2.2.1.8 Pogon za pogled Razor

Pogon za pogled Razor [18] nam pomaga pri prikazovanju pogledov. Z uporabo označevalne sintakse jezika Razor lahko vstavimo strežniško kodo na spletne strani, ki se nato pretvori v ustrezno HTML kodo.

Razor podpira strogo tipiziranje. Krmilniki lahko posredujejo strogo tipizirane modele pogledu in na ta način omogočijo preverjanje tipov ter podporo urejevalniku kode IntelliSense.

Ena izmed novih možnosti so pomožne značke Tag Helpers, ki nam omogočijo pisanje kode Razor s sintakso, ki je bližje jeziku HTML.

2.2.2 C#

C# [19] je večparadigmski programski jezik, ki obsega močno tipizacijo in imperativno, deklarativno, funkcijsko, generično, komponentno ali objektno usmerjeno programiranje. . Obnavljanje pomnilnika se izvaja z avtomatskim sproščanjem pomnilnika.

C# se pri skladnji zgleduje po številnih drugih programskih jezikih, najbolj izrazito po C, C++ in Javi. Jezik je bil skrbno načrtovan z namenom, da bi odpravil pomanjkljivosti, ki so se pojavile pri drugih programskih jezikih.

Zaradi točne specifikacije programskega jezika in vmesnega jezika se lahko C# program neposredno prevede in izvede v drugih okoljih, ne da bi bilo treba program spreminjati. C# je

razvilo podjetje Microsoft v okviru razvoja ogrodja .NET. Zadnja različica programskega jezika je označena s C# 6.0.

2.2.3 HTML

HTML [20] je označevalni jezik za nadbesedila, ki se uporablja za izdelavo spletnih strani. Predstavlja osnovo spletnega dokumenta. Poleg prikaza v spletnem brskalniku se z njim določi tudi zgradba in semantični pomen delov dokumenta.

2.2.4 CSS

CSS [21] so kaskadne slogovne podloge. Predstavljene so v obliki preprostega slogovnega jezika, ki skrbi za predstavitev spletnih strani. Z njimi definiramo slog HTML in XHTML dokumentov preko določitve pravil za prikaz posameznih elementov na spletni strani. Določamo lahko barve, velikosti, odmike, poravnave, obrobe, pozicije in vrsto drugih atributov. Prav tako pa lahko nadziramo aktivnosti, ki jih uporabnik izvaja nad elementi strani. Podloge so bile razvite z namenom, da bi dobili usklajen način podajanja informacij o slogu spletnih dokumentov.

2.2.4.1 Ogrodje Bootstrap in zbirka slogovnih predlog Bootswatch

Bootstrap [22] je brezplačno prosto dostopno programsko ogrodje, ki je namenjeno oblikovanju spletnih strani in spletnih aplikacij. Pobuda za nastanek ogrodja je nastala znotraj podjetja Twitter, ki je želelo poenoten način za razvoj grafičnih vmesnikov. Poleg mnogih stilskih predlog za tipografijo, obrazce, gumbe in navigacijo omogoča tudi preprosto prilagajanje sloga ter postavitev elementov glede na velikost zaslona naprave.

Bootswatch [23] je zbirka prosto dostopnih slogovnih predlog za ogrodje Bootstrap. Uporabna je za hitro spreminjanje videza grafičnega vmesnika.

2.2.5 JavaScript

JavaScript [24] ali krajše JS je lahek, objektno usmerjeni jezik, ki funkcije obravnava na enak način kot preostale spremenljivke. Poznamo ga predvsem kot skriptni jezik v spletnem okolju, čeprav se lahko uporablja tudi izven njega. Je dinamičen skriptni jezik, ki podpira prototipni način programiranja in podpira več modelov programiranja kot so na primer objektno usmerjeno programiranje, imperativno programiranje ter funkcijsko programiranje.

JS teče na strani odjemalca in upravlja obnašanje spletnih strani ob določenih dogodkih.

2.2.5.1 Knjižnica jQuery

jQuery [25] je hitra in majhna knjižnica spisana v jeziku JavaScript, ki nam omogoča bogat nabor zmožnosti kot so recimo obhod po elementih HTML dokumenta, manipuliranje z njegovimi elementi, upravljanje dogodkov, animacijo ter olajša delo s tehnologijami AJAX.

Uporabili smo več vtičnikov, ki razširjajo možnosti knjižnice jQuery. Vtičnik jQuery Validation [26] nam omogoča preverjanje veljavnosti vrednosti v obrazcih že na strani uporabnika. Vsebuje mnogo metod, (kot je na primer preverjanje pravilnosti zapisa elektronskega naslova in elektronske pošte) dopušča pa nam tudi možnost dodajanja in spreminjanja obstoječih metod po meri. Knjižnica jQuery Unobtrusive Validation [27] dopolnjuje prej omenjeni vtičnik jQuery Validation tako, da lahko namesto klicev funkcij za preverjanje dodamo kar atribut `data-*` znotraj HTML elementa. Tako dobimo isto funkcionalnost, a lažjo berljivost HTML kode.

2.2.5.2 Knjižnici Highcharts JS in Highstock JS

Knjižnica Highcharts JS [28] je napisana v jeziku JavaScript in nam omogoča enostavno dodajanje interaktivnih grafov na spletno stran ali v spletno aplikacijo. Ko so razvijali knjižnico, so imeli v mislih mobilne naprave, zato so grafi odzivni na večprstno prilagajanje oddaljenosti in vsebujejo prstom enostavno dostopne zaslonske namige. V novejših brskalnikih se grafi prikazujejo s pomočjo označevalnega jezika SVG [29], v starejših pa s pomočjo označevalnega jezika VML [30]. Knjižnica podpira več vrst grafikonov.

Highstock JS [28] je programska rešitev, ki je specializirana za grafikone z velikim številom podatkov na časovnici. Vse podane podatke zbere v grafikon, ki omogoča enostavno izbiranje časovnega razpona, odčitavanja vrednosti po časovni enoti in izvoz v zapise PNG, JPEG, PDF ter SVG.

Poglavje 3 Razvoj spletne aplikacije

V tem poglavju bomo predstavili razvoj aplikacije. Opisane bodo vse faze od zajema zahtev, priprave prototipa za prikaz uporabniku, programiranja aplikacije in testiranja do demonstracije delovanja.

Postopek razvoja je bil prepuščen izbiri razvijalca te spletne aplikacije. Pri razvoju se sicer nismo v celoti držali nobene izmed uveljavljenih metodologij za razvoj programske opreme. Uporabili smo princip prototipiranja, vendar bi lahko rekli, da smo razvijali predvsem v skladu s principi agilnosti [31]: hiter razvoj enostavne rešitve z aktivno udeležbo uporabnika, omejena dokumentacija ipd.

3.1 Zbiranje zahtev uporabnika

Prva razvojna aktivnost diplomske naloge je zbiranje zahtev. V ta namen je potekal sestanek s ciljnim uporabnikom in informatikom v poslovni enoti podjetja CPG, d. d., asfaltna baza Laže. Ciljni uporabnik je izpostavil problem neizkoriščenosti podatkov, odčitanih s senzorjev v gorilniku, ki se trenutno hranijo le v obliki tabele (slika 3.1) v podatkovni bazi in ostajajo neizkoriščeni za mogoče druge uporabe.

Slika 3.1: Načrt tabele Gorivo.

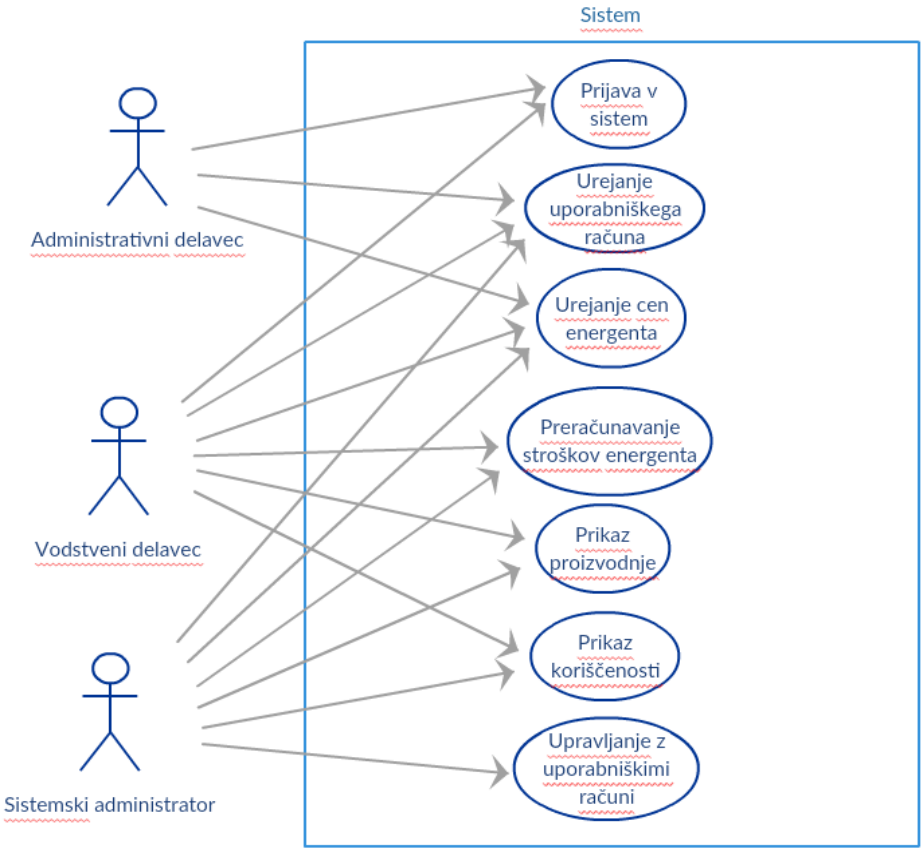
Začetna ideja je bila, da se zgradi namizno aplikacijo s preglednim uporabniškim vmesnikom, ki bi omogočala vpogled v te podatke tudi za uporabnike, ki nimajo dostopa do podatkovne baze oziroma so nevešči upravljanja z njo. Poleg tega bi omogočili še vnos cene energenta in izračun stroškov za določeno obdobje glede na ceno.

Po razmisleku in pogovoru z mentorjem smo se odločili idejo realizirati drugače. Namesto namizne aplikacije naj bi tako izdelali namensko spletno aplikacijo, ki omogoča prilagajanje prikaza glede na značilnosti zaslona, na katerem se prikazuje. Spletna aplikacija naj bi vsebovala tudi primeren nivo varnosti in kontrole nad dostopi ter druge osnovne funkcionalnosti spletnih aplikacij, recimo upravljanje z uporabniškim računom. Za razvoj same aplikacije je bilo izbrano ogrodje ASP.NET kot razširjena in uveljavljena tehnologija za razvoj sodobnih spletnih aplikacij.

Zahteve sem podrobno analiziral in uredil v dokument Specifikacija zahtev programske opreme. Ko je bil dokument potrjen s strani podjetja in s strani mentorja, smo prešli na naslednjo fazo razvoja.

3.2 Načrtovanje

Ker je spletna aplikacija namenjena uporabnikom znotraj podjetja, je večina funkcionalnosti nedostopna neprijavljenim uporabnikom. Nekatere funkcionalnosti so dostopne le uporabnikom z določeno vlogo (stopnjo pravic). Imamo tri vloge: administrativni delavec, vodstveni delavec in sistemski administrator. Administrativni delavec ima osnovno stopnjo pravic, ki mu omogoča prijavo v sistem, urejanje uporabniškega računa in urejanje cen energenta. Več pravil ima vodstveni delavec, ki poleg osnovnih pravic lahko še preračunava stroške energenta, prikazuje izplen proizvodnje in pregleduje časovni delež uporabe gorilnika. Sistemski administrator ima dostop do vseh naštetih funkcionalnosti, poleg tega pa lahko še do upravlja z uporabniškimi računi. Slika 3.2 prikazuje diagram primerov uporabe, ki ustrezajo funkcionalnostim določene stopnje pravic.

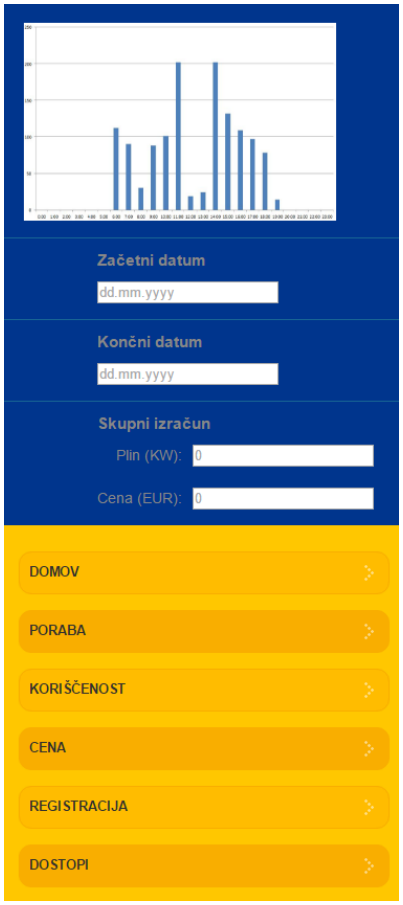


Slika 3.2: Diagram primerov uporabe za določeno vlogo.

3.3 Priprava prototipov

Poleg prvotnega namena pridobivanja odziva uporabnika je bil namen razvoja prototipov tudi postopno spoznavanje s tehnologijami za razvoj spletnih aplikacij.

Za prvi prototip smo uporabili tehnologije HTML in CSS, sam razvoj prototipa pa je potekal v orodju Notepad++. Prvi prototip je bil namenjen le preverjanju določenih delov uporabniškega vmesnika in je zajemal po namenih ločene podstrani s pripadajočimi gradniki. Slike grafov so bile statične (slika 3.3). Omogočeno je bilo prehajanje med zasloni preko navigacijskega menija, prikaz strani pa se je samodejno prilagajal širini okna.



Slika 3.3: Prototip uporabniškega vmesnika.

Izdelava nadaljnjih prototipov je bila namenjena tudi raziskovanju zalednih tehnologij oziroma postopnemu dodajanju in testiranju funkcionalnosti. Ker so prototipi nastajali le za potrebe razvoja, jih na tem mestu ne bomo podrobneje opisovali.

3.4 Izbira tehnologij

V tem podpoglavju bomo opisali korake, ki so vodili k izbiri ustreznih tehnologij. Ti so kasneje omogočili učinkovit razvoj končne rešitve.

3.4.1 Izbira med ASP.NET Web Forms in ASP.NET MVC

ASP.NET Web Forms [32] je del ogrodja ASP.NET in eden izmed modelov za izdelavo spletne aplikacije. Posamično spletno stran v tem modelu pojmujemo kot spletni obrazec. Zato je zgrajena kot kombinacija tehnologije HTML, gradnikov uporabniškega vmesnika (poimenovanih kontrole) in kode na strani strežnika. Vsak spletni obrazec ima lasten življenjski cikel, ki se odraža kot zaporedje dogodkov od inicializacije do odstranitve. Po

zahtevku uporabnika se stran prevede in izvede najprej na strani strežnika, nato pa se pretvori v ustrezno kodo HTML, ki je posredovana uporabniku.

Oblikovanje je poenostavljeno, ker imamo na voljo veliko izbiro že izdelanih komponent in čarovnikov, ki vodijo do delnega izdelka kar na podlagi izbire parametrov. Za razvoj na strani strežnika imamo na voljo programska jezika Visual Basic in C#.

Zaradi opisanih prednosti smo zato najprej poskušali izdelati aplikacijo po modelu Web Forms, ampak smo to možnost kasneje opustili in se osredotočili na model MVC, ki je bil predstavljen v poglavju 2. Pri odločitvi je pretehtalo boljše razumevanje problema in posledična nujnost uporabe tega modela. Nova verzija ogrodja MVC obljublja tudi napredek pri enostavnosti grajenja spletnih aplikacij in hkrati sledenje modernim smernicam pri razvoju spletnih tehnologij.

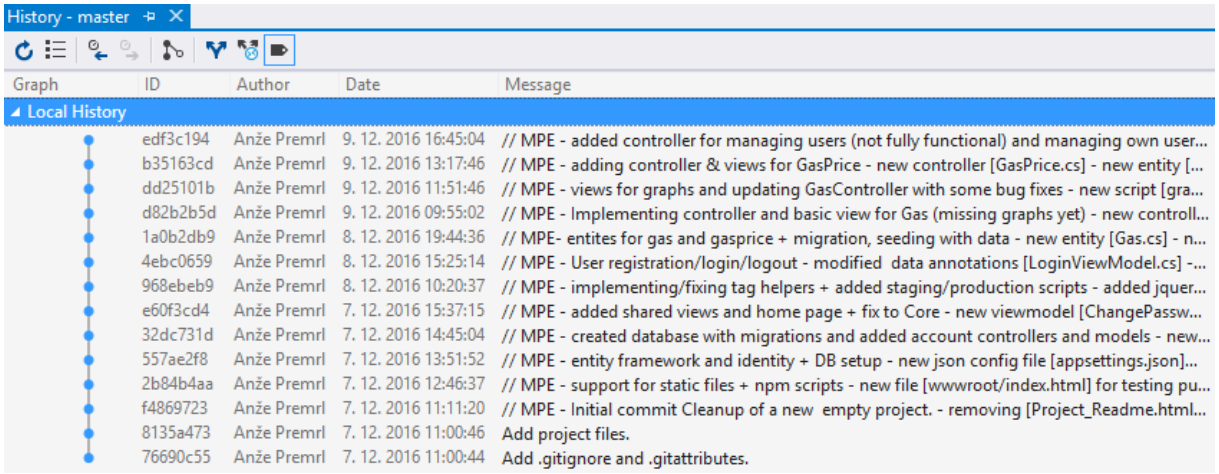
3.4.2 Izbira knjižnice za grafikone

Izbiri knjižnice za grafikone Highcharts JS je botrovala izkušnja s pomanjkljivimi zmožnostmi privzetih grafikonov, ki so že na voljo v ogrodju ASP.NET. Pri izgradnji prototipa aplikacije po metodi spletnih obrazcev smo preizkusili tudi knjižnico DotNet.Highcharts [33], ki olajša vstavljanje grafikonov v spletne strani pri razvoju v ogrodju ASP.NET. Zaradi nekompatibilnosti z ASP.NET Core smo jo morali opustiti, zato smo grafikone izdelali s pomočjo jezika JavaScript.

3.5 Razvoj končne spletne aplikacije

V tem poglavju bomo opisali, kako je potekal razvoj končne različice spletne aplikacije. Že pred tem delom razvoja je bilo pri testiranju različnih tehnologij in izgradnji zahtevanih funkcionalnosti izdelanih več prototipov, pri čemer smo si nabrali dragocene izkušnje. Prav tako smo si pomagali z zgledi delujočih aplikacij in predlog, ki so na voljo v orodju Visual Studio Community 2015 [7] [8].

Posamezni razdelki ustrezajo korakom, ki so običajno sestavljeni iz programiranja, testiranja in objavljanja sprememb v sistem za nadzor različic Git [9]. Slika 3.4 prikazuje zgodovino objavljenih sprememb pri trenutni različici spletne aplikacije.



Slika 3.4: Pregled zgodovine objavljenih sprememb.

3.5.1 Nova rešitev

Pri tvorbi nove rešitve znotraj orodja Visual Studio Community 2015 izberemo naslednje možnosti:

- predloga ASP.NET Core Web Application (.NET Core),
- izberemo možnosti za tvorbo imenika za rešitev in imenika za odlagališče za potrebe sistema nadzor različic Git,
- v naslednji maski izberemo prazno predlogo in potrdimo možnost brez preverjanja istovetnosti identitete.

3.5.2 Čiščenje predloge

Samodejno ustvarjena rešitev ima nekatere za nas nebistvene elemente, zato jo lahko delno očistimo:

- odstranimo pozdravno spletno stran,
- odstranimo še neuporabljene uvoze imenskih prostorov in komentarjev kode v razredih `Program` in `Startup`.

3.5.3 Podpora za streženje statičnih vsebin

Ogrodje ASP.NET Core je zasnovano na modularnosti, zato moramo za dodatne funkcionalnosti dodati ustrezne module v obliki paketov NuGet. Podporo za statične vsebine na strežniku zagotovimo tako, da:

- dodamo sklic na modul vmesne programske opreme `Microsoft.AspNetCore.StaticFiles` v datoteko `project.json`,
- dodamo sklice na vmesno programsko opremo v razred `Startup`,
- dodatno razširimo možnost streženja statičnih vsebin izven imenika `wwwroot` z lastnim razredom `ApplicationBuilderExtensions`.

Dodajanje vmesne programske opreme poteka na zgoraj opisan način, zato v naslednjih korakih ni posebej omenjan.

3.5.4 Kreiranje podatkovne baze in podpora za članstvo

Za preslikovanje med objekti .NET in entitetami v bazi uporabimo ogrodje Entity Framework Core. Sistem ASP.NET Core Identity nam omogoča upravljanje s članstvom, prijavljanje v aplikacijo, upravljanje s podatki uporabnikov in druge sorodne možnosti.

Ustvarimo razred `MpeDbContext`, ki predstavlja kontekst podatkovnega modela za našo podatkovno bazo. Z uporabo ustreznih ukazov ustvarimo podatkovno bazo na podlagi prej navedenega konteksta.

3.5.5 Krmilniki in ustrezni modeli pogleda

V tem koraku dodamo ogrodje Asp.Net Core MVC, ki je nujno potrebno pri nadaljnjih korakih. Dodamo tudi krmilnik, ki bo vseboval vso logiko, ki je povezana z uporabniškimi računi, krmilnik za domačo stran in modela pogleda za registracijo ter prijavo.

3.5.6 Osnovni pogledi in slog uporabniškega vmesnika

V datoteko `package.json` (to je datoteka z nastavitvami upravljalnika s paketi Node.js), dodamo vnos za Bootswatch in ustvarimo lastno slogovno datoteko `site.css`. Tu lahko vidimo kratek del vsebine datoteke `package.json`.

```
{"version": "1.0.0",
"name": "mpe",
"private": true,
"devDependencies": {
       "bootstrap": "3.3.7",
       "bootswatch": "3.3.7",
       "jquery": "3.1.1",
…},}
```

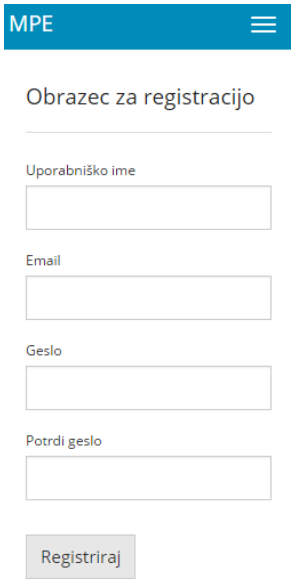
Dodamo tudi vse poglede, ki poimensko in lokacijsko ustrezajo konvenciji (`ViewStart`, `_ViewImports`, `_Layout`) ter poglede za potrebe aplikacije.

3.5.7 Pomožne značke in delni pogled s skriptami za validacijo

Ustvarimo delni pogled `_ValidationScriptsPartial`, ki vsebuje povezave na skripte, ki so potrebne za preverjanje pravilnosti vnosov v obrazcih. Ta delni pogled bomo po potrebi vdelali v poglede, ki potrebujejo njegove zmožnosti. Dodamo tudi podporo za pomožne značke.

3.5.8 Pogledi za registracijo in prijavo uporabnika

Ustvarimo še poglede za registracijo (slika 3.5) uporabniškega računa ter prijavo v aplikacijo. Tako imamo to funkcionalnost do neke mere že delujočo, v nadaljnjih korakih pa jo bomo še ustrezno dopolnili.



Slika 3.5: Obrazec za registracijo uporabnika.

3.5.9 Entitete za plin in ceno plina

Dodamo modele za plin in ceno plina, ki predstavljajo entitete le-teh v podatkovni bazi. Ustvarimo tudi razred, ki bo skrbel za polnjenje baze iz priskrbljenih virov. Trenutno je nastavljen tako, da prebere podatke datotek v ustreznem formatu jezika JSON [34]. Ustrezen format je viden na odseku datoteke `gas.json`, ki ustreza enemu vnosu v tabelo `Plin`.

```
[…, {
       "stevilka_vpisa": 23,
       "datum": "2013-05-31T00:00:00",
       "ura_zacetnega_vpisa": "1899-12-30T05:46:56",
       "ura_koncnega_vpisa": "1899-12-30T18:40:12",
       "zacetno_stanje": 853563,
       "koncno_stanje": 854681,
       "proizvedena_kolicina": 146.94,
       "zacetno_stanje_cela": 13,
       "zacetno_stanje_cifra": 15955,
       "koncno_stanje_cela": 13,
       "koncno_stanje_cifra": 27135
}, …]
```

3.5.10 Prikazovanje vnosov v tabelo Plin za izbrano časovno obdobje

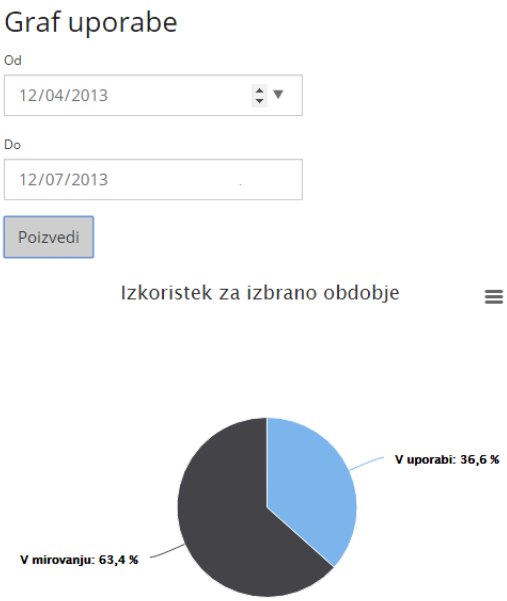
Ustvarimo krmilnik in poglede za potrebe prikazovanja vnosov iz tabele `Plin` (slika 3.6) za izbrano časovno obdobje. Prav tako ustvarimo servisni razred `DateToLong`, ki nam pomaga

pri pretvorbi datumov v ustrezno številsko vrednost, ki je potrebna za prikaz grafikonov Highstock JS.

Slika 3.6: Prikaz izračuna porabe in vnosov za izbrano časovno obdobje.

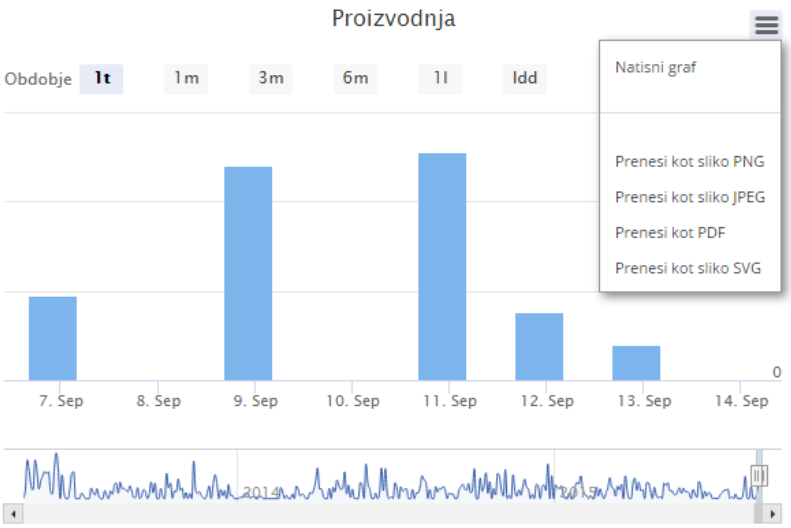
3.5.11 Prikazovanje grafikonov

V tem koraku dodamo pogled za grafikone in delni pogled za krožni grafikon, ki prikazuje izkoristek (slika 3.7). Dodamo knjižnico Highcharts JS in ustvarimo skripti za grafikone, ki ju vključimo v prej omenjene poglede.



Slika 3.7: Prikaz delež uporabe gorilnika za izbrano časovno obdobje.

Sami grafikoni dobijo podatke za prikaz preko akcij ustreznega krmilnika v zapisu JSON. Poleg izrisa v spletni aplikaciji je omogočeno tudi tiskanje ali prenos grafikona na odjemalca (slika 3.8).



Slika 3.8: Možnost prenosa grafikona v različnih formatih.

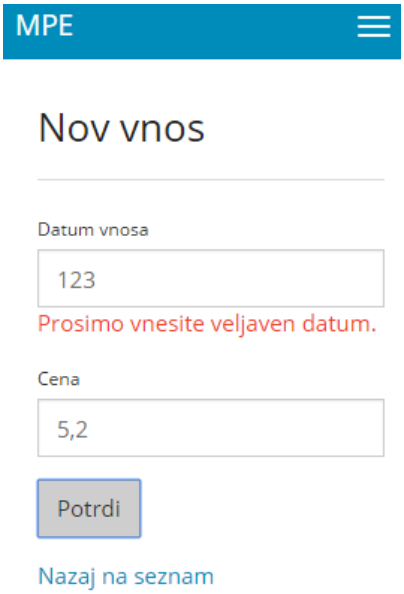
3.5.12 Entiteta, krmilnik in pogledi za ceno

S pomočjo tehnike ASP.NET Scaffolding [35] lahko samodejno generiramo akcije in poglede, ki ustrezajo operacijam CRUD. To tehniko smo uporabili pri izdelavi prototipa, za to različico pa ne, ker smo predhodno že naredili nekatere spremembe, ki jih nismo želeli prepisati.

Poleg tega smo morali za pravilen vnos in urejanje števil z decimalno vejico razširiti knjižnico jQuery Validation s skripto, ki je podrla lokalizirano obliko zapisa. Kot je vidno na spodnjem primeru skripte smo uporabili nemško obliko zapisa datuma ter številk, ki ustreza tudi potrebam slovenskega jezika.

```
$.extend($.validator.methods, {
	date: function (value, element) {
		return this.optional(element) ||
			/^\d\d?\.\d\d?\.\d\d\d?\d?$/.test(value);
	},
	number: function (value, element) {
		return this.optional(element) || /^-
			?(?:\d+|\d{1,3}(?:\.\d{3})+)(?:,\d+)?$/.test(value);
	}
});
```

Slika 3.9 prikazuje preverjanje pravilnosti novega vnosa cene za energent.



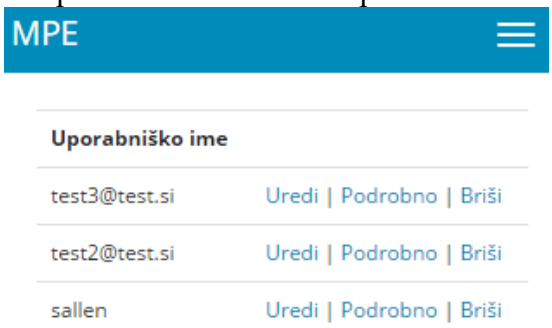
Slika 3.9: Preverjanje pravilnosti vnosa pri vnosu cene energenta.

3.5.13 Upravljanje z uporabniškimi računi

V tem koraku smo dodali funkcionalnost za upravljanje uporabniškega računa in upravljanje uporabniških računov s strani administratorja. Podobno kot v prejšnjem koraku, smo predhodno uporabili tehniko ASP.NET Scaffolding in prilagodili ustvarjene akcije ter poglede po lastni potrebi. V nadaljevanju predstavljena metoda `ChangePassword` recimo omogoča registriranim uporabnikom spreminjanje uporabniškega gesla.

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> ChangePassword(ChangePasswordViewModel model) {
       if (!ModelState.IsValid) {
              return View(model);
       }
       var user = await GetCurrentUserAsync();
       if (user != null) {
              var result = await _userManager.ChangePasswordAsync(user,
              model.OldPassword, model.NewPassword);
              if (result.Succeeded) {
                     await _signInManager.SignInAsync(user, isPersistent: false);
                     logger.LogInformation(3, "Uporabnik je uspešno spremenil
                     uporabniško geslo.");
                     return RedirectToAction(nameof(Index), new { Message =
                     ManageMessageId.ChangePasswordSuccess });
              }
              AddErrors(result);
              return View(model);
       }
       return RedirectToAction(nameof(Index), new { Message = ManageMessageId.Error });
}
```

Slika 3.10 prikazuje seznam uporabniških računov in povezave do akcij nad njimi.

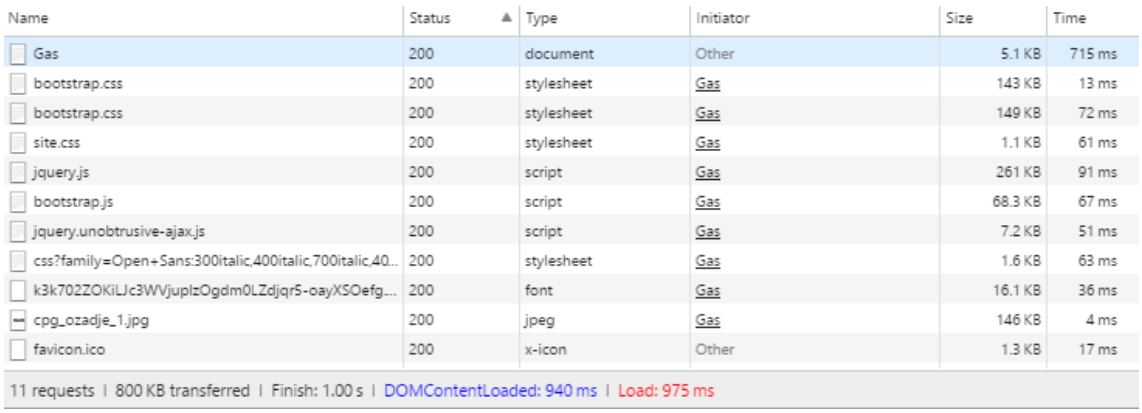


Slika 3.10: Seznam uporabniških računov v meniju administratorja.

3.6 Testiranje in namestitev

Za potrebe razvoja in testiranja smo spletno aplikacijo poganjali na osebnem računalniku s 64 bitnim sistemom Windows 10 Pro, procesorjem Intel Core i3-6100 z delovnim taktom 3,7 gigahertzov, z osmimi gigabajti delovnega pomnilnika, diskom SSD in z uporabo privzetega strežnika za ASP.NET Core poimenovanega Kestrel [36].

Za odkrivanje in popravljanje napak v programski kodi smo uporabljali razhroščevalnik, ki je na voljo znotraj orodja Visual Studio Community 2015 [7], [8] ter orodje Chrome Developer Tools [10]. Slednjega smo uporabljali tudi pri meritvah časa nalaganja strani (slika 3.10).



Slika 3.10: Prikaz zajema podatkov omrežja z uporabo orodja Chrome Developer Tools.

Testiranje smo opravljali korak za korakom ob dodajanju določene funkcionalnosti. Poleg preverjanja pravilnosti delovanja in stabilnosti aplikacije smo sproti testirali še uporabniški vmesnik pri različnih velikostih okna brskalnika. Opazovali smo tudi obnašanje strani v različnih brskalnikih: Google Chrome, Mozilla Firefox, Microsoft Edge Web Browser in Internet Explorer 11.

Same aplikacije nismo še namestili v končno delovno okolje. Uporabnik je testiral njeno delovanje na računalniku, na katerem je potekal razvoj. Namestitev je prepuščena nadaljnjemu sodelovanju med razvijalcem in uporabnikom. S tem zaključujemo pregled razvoja aplikacije. V naslednjem poglavju bomo predstavili analizo opravljenega dela in ovrednotili končno različico aplikacije.

Poglavje 4 Analiza rešitve

V tem poglavju bomo predstavili odziv uporabnikov aplikacije ter ovrednotili opravljeno delo razvijalca. Zadnja različica rešitve vsebuje vse predvidene funkcionalnosti, pri čemer smo nekatere funkcionalnosti ustrezno prilagodili. Predvsem pa smo postavili ogrodje, ki bo olajšalo prihodnje prilagoditve in nadgradnjo.

4.1 Odziv uporabnikov

V tem razdelku bomo predstavili odziv uporabnikov na izgled uporabniškega vmesnika in odziv na razporeditev funkcionalnosti, ki so dostopne uporabnikom z različnimi vlogami v sistemu.

4.1.1 Uporabniški vmesnik

Aplikacija je oblikovana tako, da se prilagaja velikosti zaslona naprave in olajša rokovanje na mobilnih napravah. Uporabnik je pohvalil izgled uporabniškega vmesnika in odzivnost spletne aplikacije. Predvsem ga je presenetila hitrost poizvedb po bazi. Za dodatne prilagoditve bi potrebovali odziv več uporabnikov, saj je trenutno kot edini iz podjetja aplikacijo uporabljal le on sam.

4.1.2 Funkcionalnosti in vloge

Uporabnik se je prijavil v vseh treh vlogah: kot administrativni delavec (prijava v sistem, urejanje uporabniškega računa ter urejanje cene energenta), vodstveni delavec (vse to ter še preračunavanje stroškov energenta, prikaz proizvodnje, prikaz časovnega deleža uporabe gorilnika) ter kot sistemski administrator, ki poleg vsega lahko upravlja še uporabniške račune. Uporaba funkcionalnosti se sklada z njegovo predstavo, zato uporabniku ni povzročala težav. Osnovne funkcionalnosti so izvedene enako kot pri podobnih aplikacijah, preostale pa so zasnovane tako, da je uporabnik z njimi ravnal intuitivno. Smiselna razporeditev vlog in njihovih funkcionalnosti v aplikaciji tudi ustreza potrebam po dostopu do informacij za zaposlene na ustreznih mestih v podjetju.

Po preizkusu vseh funkcionalnosti in vlog je uporabnik ocenil, da smo uspeli zadostiti začetnim zahtevam in dosledno upoštevali odziv iz preteklih preizkusov. Prav tako je dobil več idej o razširitvah obstoječih ali dodajanju novih funkcionalnosti. Nekatere izmed teh bomo omenili v sklepnem poglavju.

4.2 Ovrednotenje dela razvijalca in ugotovitve

Za razvoj spletne aplikacije smo porabili približno osem tednov. Od tega smo štiri tedne potrebovali za učenje tehnologij, en teden smo porabili za razvoj rešitve zgrajene po modelu ASP.NET Web Forms [32], dva tedna za razvoj po arhitekturnem načelu MVC z uporabo ogrodja ASP.NET Core [15] in še dodaten teden za kasnejše popravke in nadgradnjo na novejšo različico ogrodja ASP.NET Core.

Spisano je približno 2400 vrstic lastne kode (mnogo večji del celotne rešitve predstavljajo uporabljene knjižnice). Od tega je 1170 vrstic v jeziku C#, 220 v jeziku Javascript, 50 v jeziku CSS in 940 vrstic kode kot kombinacija HTML in označevalne sintakse jezika Razor [18].

4.2.1 Zmogljivost

Prva odzivnost po zagonu strežnika zahteva približno 2000 milisekund, nadaljnja pa nekaj 10 milisekund. Za prikaz poizvedbe enega vnosa v bazi, ki ustreza podatkom enega dneva potrebujemo 16 milisekund, za prikaz vnosov enega meseca potrebujemo 64 milisekund, za prikaz vnosov enega leta potrebujemo 261 milisekund in 600 milisekund za vse zbrane vnose v testni bazi za obdobje, ki znaša skupno dve leti in štiri mesece.

Verodostojnost rezultatov smo zagotovili tako, da smo onemogočili predpomnjenje datotek v brskalniku. Ko bo aplikacija nameščena na namenskem strežniku bomo morali upoštevati še čas, ki ga bo zahtevek porabil za pot od uporabnika do strežnika po medmrežju ter nazaj.

4.2.2 Težave med razvojem

Največ težav pri razvoju je bilo povezanih z zrelostjo programskega ogrodja ASP.NET Core, ki je bil v času razvoju še v eni izmed različic brez dolgoročne podpore. To se je poznalo predvsem v manjšem naboru funkcionalnosti v primerjavi stabilno različico ogrodja ASP.NET različice 4.6. Za slednjo je na voljo tudi več gradiva, primerov in podpore s strani obstoječih uporabnikov ogrodja.

4.2.3 Ugotovitve

Ker smo funkcionalnosti dodajali postopoma, iz prototipa v prototip, smo na koncu ugotovili, da bi lahko mnogo negotovosti in improvizacije prihranili z predhodno izdelavo podrobno definiranih uporabniških vmesnikov. Prav tako bi lahko izvedbo podkrepili z uporabo testno naravnanega razvoja programske opreme. Zadovoljni pa smo z izbiro programskega ogrodja, orodij in tehnologij, ki so aktualne [32] in po katerih vlada veliko povpraševanje na trgu dela [37]. Kot kakovostno alternativo bi lahko uporabili programsko ogrodje MEAN [38]. V tem primeru so vse komponente (tako za obličje kot za vsa zaledja) v celoti zgrajene v jeziku Javascript.

Poglavje 5 Sklepne ugotovitve

Cilj diplomskega dela je bila seznanitev z vsemi fazami izdelave informacijske rešitve v obliki spletne aplikacije za problem, ki izvira iz stvarnosti. V samem procesu smo se seznanili z mnogimi postopki, ki smo jih teoretično obravnavali pri študiju. Prav tako smo z izdelavo poleg obnovitve in pridobitve novih znanj dobili tudi priložnost s tem znanjem ustvariti nekaj celovitega. Uspelo nam je torej razviti namensko spletno aplikacijo v okviru podanih zahtev, ki je zgrajena po sodobnih smernicah in je dobro izhodišče za nadaljnje razširitve.

Čeprav aplikacija trenutno še ni umeščena v delovno okolje, je bodoči uporabnik zadovoljen z dosedanjim delom, saj bi mu taka aplikacija potencialno prihranila čas, ki ga trenutno porabi s preračunavanji s kalkulatorjem in pisalom. Poleg tega pa bo s podatki razpolagal kadarkoli in kjerkoli s kakršnekoli naprave, ki ima nameščen standardni spletni brskalnik.

Pri testiranju smo dobili več idej tako o potencialni razširitvi aplikacije izven okvirjev preračunavanja porabe energentov v gorilniku sušilnega bobna kot o izboljšavi obstoječih funkcionalnosti. Nekatere izmed predlaganih izboljšav in razširitev so:

- uporaba izdelanih funkcionalnosti tudi za porabo električne energije na nivoju poslovne enote,
- primerjava stroškov za izbrano časovno obdobje med preteklimi in aktualnimi cenami energentov,
- preračunavanje urne proizvodnje za določen dan,
- preračunavanje porabe energenta na tono proizvoda,
- primerjava proizvedene količine z meritvami na tehnološki tehnici (odprema),
- prikazovanje vsote glede na merilo grafikona (dnevne v tedenskem merilu, tedenske v mesečnem merilu, mesečne v letnem merilu).

Zaradi modularne zasnove bi izdelano rešitev lahko v bodoče nadgradili na podlagi podanih predlogov, da bi postala uporabna tudi v drugih podjetjih iz iste panoge.

Med procesom razvoja spletne aplikacije sem spoznal, da je na področju informatizacije poslovnih procesov še veliko neizkoriščenih priložnosti, kar kaže na perspektivo poklica inženirja računalništva in informatike. Največje spoznanje, ki sem ga odnesel med programiranjem rešitve je to, da ti na prvi pogled še tako veliki hrošči ne smejo vzeti poguma pri iskanju rešitve. Pravo razmerje vztrajnosti (in počitka) slej ko prej vodi do zmage.

Literatura

[1] Modbus FAQ. *Modbus Organization, Inc.* [Online]. Dosegljivo: http://www.modbus.org/faq.php. [Dostopano 12. 12. 2016].

[2] Meritve porabe energije. *Top Teh d. o. o.* [Online]. Dosegljivo: http://topteh.si/storitve/meritve_porabe_energije/. [Dostopano 17. 2. 2017].

[3] Merjenje porabe. *Elektro Maribor d. d.* [Online]. Dosegljivo: http://www.elektro-maribor.si/index.php/ucinkovita-raba/120-merjenje-porabe. [Dostopano 17. 2. 2017].

[4] About engage. *Efergy Engage Ltd.* [Online]. Dosegljivo: https://engage.efergy.com/content/about-engage. [Dostopano 17. 2. 2017].

[5] Products. *Current Cost Ltd.* [Online]. Dosegljivo: http://www.currentcost.com/products.html. [Dostopano 17. 2. 2017.].

[6] I. Golob in T. Welzer. Arhitekture podatkovnih skladišč. (2001). *Slovensko društvo Informatika* [Online]. Dosegljivo: http://www.drustvo-informatika.si/fileadmin/dsi2001/sekcija_a/golob_welzer.doc. [Dostopano 17. 1. 2017].

[7] Microsoft Visual Studio. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio. [Dostopano 15. 1. 2017].

[8] Free IDE and Tools. *Microsoft* [Online]. Dosegljivo: https://www.visualstudio.com/vs/community/. [Dostopano 15. 1. 2017].

[9] Git. *Software Freedom Conservancy* [Online]. Dosegljivo: https://git-scm.com/. [Dostopano 13. 2. 2017].

[10] Chrome DevTools Overview. *Google* [Online]. Dosegljivo: https://developer.chrome.com/devtools. [Dostopano 13. 12. 2016].

[11] T. Patton. Chrome add-on Postman streamlines testing APIs. *TechRepublic* [Online]. Dosegljivo: http://www.techrepublic.com/article/chrome-add-on-postman-streamlines-testing-apis/. [Dostopano 13. 12. 2016].

[12] K. Cook in L. Urlich Fuller, Access 2016 For Dummies, 1. izd., Hoboken: For Dummies, 2015.

[13] Microsoft Jet Database Engine. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Microsoft_Jet_Database_Engine. [Dostopano 13. 12. 2016].

[14] D. Roth, R. Anderson in S. Luttin. (oktober 2016). Introduction to ASP.NET Core. *Microsoft* [Online]. Dosegljivo: https://docs.microsoft.com/en-us/aspnet/core/. [Dostopano 15. 1. 2017].

[15] A. Freeman, *Pro ASP.NET Core MVC: Develop cloud-ready web applications using Microsoft's latest framework, ASP.NET Core MVC*, 6. izd., New York: Apress, 2016.

[16] ASP.NET Web API. *Microsoft* [Online]. Dosegljivo: https://www.asp.net/web-api. [Dostopano 8. 3. 2017].

[17] NuGet Gallery. .Net Foundation [Online]. Dosegljivo: https://www.nuget.org/. [Dostopano 8. 3. 2017].

[18] T. Mullen in R. Anderson. (januar 2017). Razor syntax. *Microsoft* [Online]. Dosegljivo: https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor. [Dostopano 14. 2. 2017].

[19] J. Sharp, Microsoft C# 2013: Step by Step Developer, 1. izd., Sebastopol: Microsoft Press, 2013.

[20] HTML. *Wikipedia* [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/HTML. [Dostopano 17. 1. 2017].

[21] CSS. *Wikipedia* [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/CSS. [Dostopano 17. 1. 2017].

[22] Bootstrap (front-end framework). *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework). [Dostopano 17. 1. 2017].

[23] T. Park. (2016). Thomaspark/bootswatch: Themes for Bootstrap. *GitHub* [Online]. Dosegljivo: https://github.com/thomaspark/bootswatch. [Dostopano 17. 1. 2017].

[24] About JavaScript – JavaScript. *Mozilla Developer Network* [Online]. Dosegljivo: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript. [Dostopano 17. 1. 2017].

[25] jQuery. *The jQuery Foundation* [Online]. Dosegljivo: https://jquery.com/. [Dostopano 17. 1. 2017].

[26] Form validation with jQuery. *jQuery Validation Plugin* [Online]. Dosegljivo: https://jqueryvalidation.org/. [Dostopano 17. 1. 2017].

[27] M. Jones. (april 2015). What is Unobtrusive Validation? - ASP.NET MVC Demystified. https://www.exceptionnotfound.net/asp-net-mvc-demystified-unobtrusive-validation/. [Dostopano 17. 1. 2017].

[28] About us. *Highcharts* [Online]. Dosegljivo: http://www.highcharts.com/about. [Dostopano 17. 1. 2017].

[29] Scalable Vector Graphics. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Scalable_Vector_Graphics. [Dostopano 17. 1. 2017].

[30] Vector Markup Language. *Wikipedia* [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Vector_Markup_Language. [Dostopano 17. 1. 2017].

[31] Manifest agilnega razvoja programske opreme. [Online]. Dosegljivo: http://agilemanifesto.org/iso/sl/manifesto.html. [Dostopano 8. 3. 2017].

[32] ASP.NET Core Killed Web Forms?. (oktober 2016). Stormpath [Online]. Dosegljivo: https://stormpath.com/blog/dotnet-core-killed-web-forms. [Dostopano 18. 1. 2017].

[33] DotNet.Highcharts. *Microsoft* [Online]. Dosegljivo: http://dotnethighcharts.codeplex.com/. [Dostopano 18. 1. 2017].

[34] Introducing JSON. *JSON* [Online]. Dosegljivo: http://www.json.org/. [Dostopano 17. 2. 2017].

[35] T. Dykstra in R. Anderson. (oktober 2016) Getting started with ASP.NET Core MVC and Entity Framework Core using Visual Studio (1 of 10). *Microsoft* [Online].

Dosegljivo: https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/intro. [Dostopano 18. 1. 2017].

[36] T. Dykstra, S. Halter in C. Ross. (oktober 2016) Kestrel web server implementation in ASP.NET Core. *Microsoft* [Online]. Dosegljivo: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel [Dostopano 23. 2. 2017].

[37] Računalništvo, programiranje. *Styria digital marketplaces, d. o. o.* [Online]. Dosegljivo: https://www.mojedelo.com/prosta-delovna-mesta/racunalnistvo-programiranje/vse-regije. [Dostopano 23. 2. 2017].

[38] Learn.mean.io. *Linnovate* [Online]. Dosegljivo: http://learn.mean.io/. [Dostopano 23. 2. 2017].